UNIVERSIDAD CATÓLICA DE TEMUCO

Taller de Integración IV: Documentación proyecto:
Asistente generador de reportes

Grupo A1

UNIVERSIDAD CATÓLICA DE TEMUCO

Introducción

Se requiere construir un módulo que permita administrar y configurar una base de datos (independiente de su MER) para generar de forma dinámica reportes SQL mediante asistentes guiados que faciliten al usuario final la construcción de sus consultas particulares que puedan ser exportados a formatos estándar de salida como Excel, PDF o JSON (para APIs).El sistema internamente conocerá cuáles son las tablas, y columnas que estarán disponibles para los reportes, permitirá internamente configurar o conocer cómo se realizan los cruces (JOINS) entre los campos seleccionados y permitirá establecer criterios para filtros, grupos o campos calculados de acuerdo a lo que el usuario requiera sin mayor conocimiento de SQL o la estructura de datos final.

UNIVERSIDAD CATÓLICA DE TEMUCO

Requerimientos del proyecto

Definición general del proyecto:

Idea general del proyecto: Construir una aplicación web que permita la construcción de reportes de manera dinámica en SQL y que estos puedan ser exportados a distintos tipos de formatos que se le necesite (Excel, PDF o JSON).

El objetivo de este proyecto es agilizar la construcción de reportes para distintos usos que se le necesite dar en la universidad y así optimizar tiempo.

Nuestra herramienta está diseñada de forma tal que no se necesita un conocimiento de ningún tipo en cuanto a consultas de base de datos, lo que hace que esté centrado en cualquier tipo de usuario que necesite dicho reporte específico.

Definición específica del proyecto

La idea es agilizar los tiempos de creación de reportes para cualquier tipo de usuario que lo necesite. Dichos reportes son:

- Reporte de gastos por carrera:
- Reportes semestral y anual
- Reportes de aprobación y reprobación de un ramo

Este proyecto no forma parte de otro preexistente, por ende ha sido desarrollado completamente de cero de forma conjunta entre los estudiantes de Taller II y Taller IV.
Tampoco existieron limitaciones para poder desarrollar el proyecto más allá de los requeridos que serán expuestos a continuación.

Herramientas utilizadas

- Github
- VUE y Bootstrap
- Laravel, SQL Server

Para el manejo de versiones de la aplicación se ha decidido utilizar Github debido a que todos ya estabamos familiarizados más con este entorno de manejo de versiones. VUE y Bootstrap al igual que Laravel con SQL Server eran solicitados en el proyecto.

Planificación

La metodología que se ha utilizado para el desarrollo del proyecto ha sido la de SCRUM, en un intervalo de 3 Sprints en los cuales se fueron presentando los avances. Cómo equipo en el primer sprint, nos centramos en crear una base para poder hacer los primeros testeos. Ya en el segundo, teniendo una base moldeable donde poder realizar los primeros reportes. Finalizando el tercer sprint con la mejora de la creación de dichos reportes, refinando las consultas.

Instalación de entorno de desarrollo

Cómo se trabajó con Laravel, es necesario la instalación de Composer, el cual es un sistema de gestión de paquetes de PHP, una vez obtenido Composer, se debe escribir en la consola el siguiente comando:

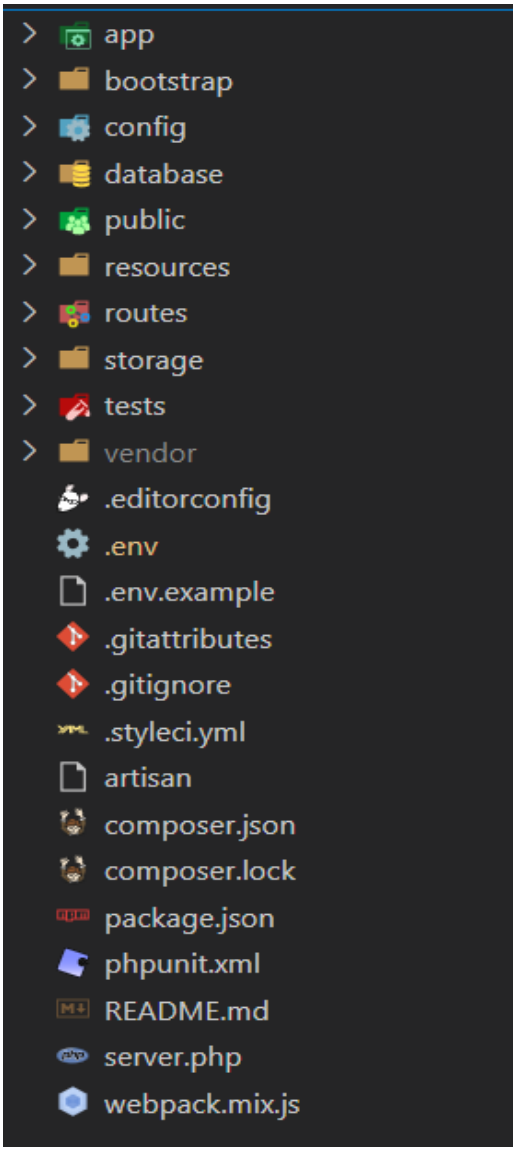
```
composer create-project laravel/laravel example-app
```

Sustituyendo “example-app” por el nombre de nuestro proyecto. Cabe destacar que se puede cambiar de versión de Laravel de esta forma:

```
composer create-project --prefer-dist laravel/laravel:^7.0
```

Siendo esta la versión de Laravel que utilizaremos.

Se nos genera la siguiente estructura la cual será la base en donde se trabajara el proyecto



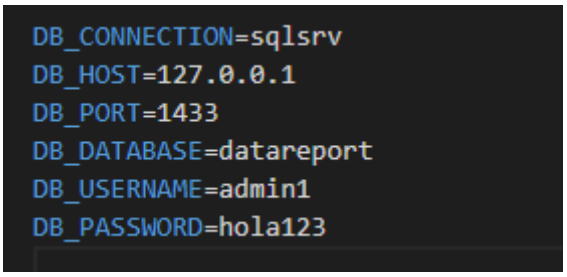
A continuación se instalará SQL Server a traves del siguiente enlace:

https://www.microsoft.com/es-es/sql-server/sql-server-downloads

También se deben descargar drivers para SQL Server, ya que sin estos genera constantes errores:
https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-ver16

Y finalizando con SQL Server, un gestor de base de datos:
https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Para crear la base de datos se puede hacer de dos formas, manual (que es nuestro caso) o con Laravel, pero para que eso ocurra primero debemos configurar el archivo .env de nuestro proyecto, siendo estas líneas de código las más importantes:



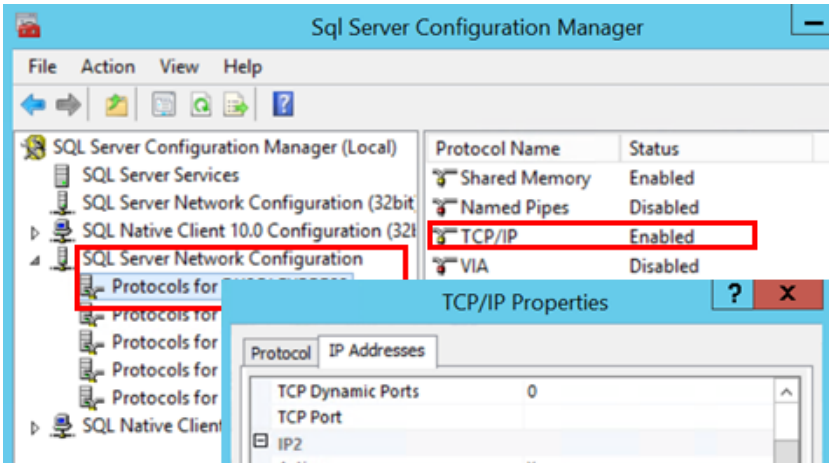
DB_CONNECTION se encarga de establecer el motor de las base de datos , DB_HOST siendo 127.0.0.1(o localhost), DB_PORT siendo 1433, esto es necesario para poder usar de manera correcta SQL Server, DB_DATABASE es el nombre de la base de datos. DB_USERNAME y DB_PASSWORD son el usuario y la password respectivamente. Estos deben ser creados anteriormente en Microsoft SQL Server Management Studio, una vez conectado dirigirse a la carpeta security -> logins -> New login(), con esto se creará un usuario y contraseña los cuales deben coincidir con lo escrito en el archivo .env.

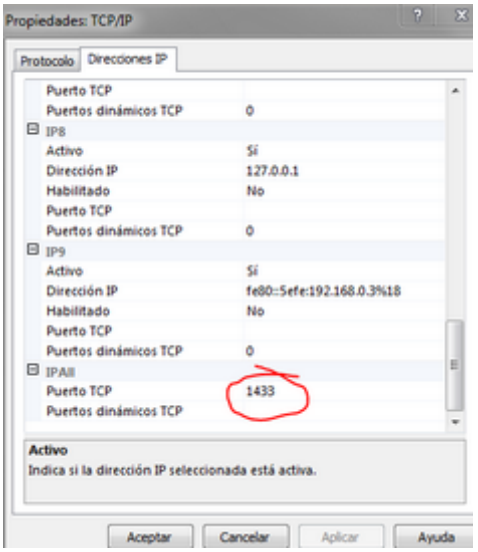
Además de todo lo mencionado anteriormente se deben añadir los archivos:

- “extension=php_pdo_sqlsrv_74_ts_x64.dll”
- “extension=php_sqlsrv_74_ts_x64.dll”

Estos archivos deben ser ubicados dentro de la carpeta “C:\xampp\php\ext\” y deben ser especificados en el archivo php.ini ubicado en la misma ruta mencionada anteriormente.

Finalmente y cómo se mencionó antes, el puerto 1433 debe estar funcionando, para esto debemos activar post instalación de SQL Server los protocolos TCP/IP:





Una vez completado estos procesos, se está listo para empezar a trabajar.

Aspectos técnicos: Código

Debido a que los códigos de Laravel son bastante extensos, sólo se pondrán las funciones más importantes dentro del proyecto
Partiendo por la parte de los controladores creados

```php
public function mostrarprofesores()
{
    $data = DB::table('Profesor')->get();
    if (Auth::check()) {
        return view('profesor', ['profesores' => $data]);
    } else {
        return redirect()->route('user.login');
    }
}
```

Cómo se observa, la función mostrarprofesores() obtiene todos los datos de la tabla Profesor a través del método get() de Laravel, siguiendo con la función, si la autenticación de usuario está funcionando y en orden, a este se le retorna a la vista profesor, pasandole los datos de la tabla profesor en la variable $data. En el caso de no tener un logueo correcto se le redirecciona al usuario a la pantalla de login.

Este proceso se repite con todas las tablas de la base de datos, solo que varia el dato (en vez de profesor pasa por alumno, ramo, departamento,etc)

Pasando por las vistas

```
@extends('layouts.master')
<html>

<head>
    <title>Tabla Alumnos</title>
</head>
@section('content')

<body class="index_main">

    <div class="table_box">
        <table id="tabla" class="table table-striped table-responsive" style="width:100%">
            <thead>
                <tr>
                    <th scope="col">RUT</th>
                    <th scope="col">Nombre</th>
                    <th scope="col">Apellido</th>
                    <th scope="col">Correo</th>
                    <th scope="col">Año de ingreso</th>
                    <th scope="col">Id de carrera</th>
                </tr>
            </thead>
            <tbody>
```

Cada vista se extiende de un html llamado master en la carpeta layout, dicho html es la base donde se encuentra la base de la aplicación. Cada tabla de la base de datos tiene su propia vista.

```
            <tbody>
                @foreach($profesores as $item)
                <tr>
                    <th scope="row">{{$item->Rut_Profesor}}</th>
                    <td>{{$item->Nombres}}</td>
                    <td>{{$item->Apellidos}}</td>
                    <td>{{$item->Correo}}</td>
                    <td>{{$item->Anho_Contrato}}</td>
                    <td>{{$item->Sueldo}}</td>
                    <td>{{$item->Contrato}}</td>
                    <td>{{$item->Id_Departamento}}</td>
                    <td>{{$item->Id_Carrera}}</td>
                </tr>
                @endforeach
```

Cómo veíamos en la primera imagen, obtenemos los datos de profesores a través de una variable $data a otra llamada profesores, simplemente con un foreach se recorre dichos datos y se van implementando según las filas de cada tabla de la base de datos.

Al igual que con las funciones, esto se repite para todas las tablas con sus respectivas columnas

También tenemos las rutas de la aplicación, el cual nos redireccionan llamando a sus respectivas funciones

```php
Route::get('/', 'UserController@index');
Route::post('/index', 'UserController@check')->name('login.check');
Route::post('/login', 'UserController@logout')->name('user.logout');
Route::get('/login', 'UserController@index')->name('user.login');
Route::get('/index', 'UserController@refresh');
Route::get('/tablaprofesores', 'TableController@mostrarprofesores');
Route::get('/uct', 'NavbarController@cambiarUCT')->name('nav.uct');
Route::get('/contacto', 'NavbarController@cambiarContacto')->name('nav.contacto');
Route::get('/tablacarrera', 'TableController@mostrarcarreras');
Route::get('/tabladepartamento', 'TableController@mostrardepartamentos');
Route::get('/tablagasto', 'TableController@mostrargastos');
Route::get('/tablaramo', 'TableController@mostrarramos');
Route::get('/tablatipocarrera', 'TableController@mostrartipocarreras');
```

De la plantilla master de la carpeta layout mencionada anteriormente, se puede destacar sobretodo:@CSRF

```php
<a class="navbar-brand" href="#"> &nbspUniversidad Catolica de Temuco</a>
@if(Auth::check())
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarText" aria-control
    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarText">
    <ul class="navbar-nav mr-auto">
        <li class="nav-item">
            <a class="nav-link" href="{{route('login.check')}}">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{route('nav.uct')}}">UCT</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{route('nav.contacto')}}">Contacto</a>
        </li>
    </ul>
    <ul class="navbar-nav ml-auto">
        <form class="nav_right" action="{{ route('user.logout') }}" method="post">
            @csrf
            <img src="{{ asset('images/user.png') }}" width="60" height="60" class="d-inline-block align-t
              {{ auth()->user()->name }}   
            <button class="btn btn-danger" type="submit"><a>Logout</a></button>
        </form>
```

Esta función de Laravel en cada formulario crea un token para cada usuario y ver si es este mismo quien está haciendo las peticiones. Es la forma de seguridad que tiene Laravel en los formularios y evitar así exploits o excesivas peticiones a la aplicación